Universidade de Coimbra
Departamento de Engenharia Informática
Sistemas Distribuídos 2017/2018

Demo #3 — Remote Method Invocation in Java

---

1. Take a look at the client and the server applications in the **hello** directory. Before starting the server you must run the `rmiregistry` application. For instance, on the command line you should prompt the following command:

```
rmiregistry 7000
```

The registry runs on port 7000 (default value is 1099). The registry is where servers of objects store the remote references that clients can look for. In this way, we can have many servers and clients that only need to know where the registry is and the name of the object of interest. After running the registry, you can run the server, and finally the client.

2. Please consider the following questions:

2.a) Which changes should you make in the previous application and in the launch of the RMI registry to make the server object available at port 6666?

2.b) The client holds a reference to some object implementing the `Hello` interface. Did we write that implementation? What kind of object is this?

2.c) Can you launch the `rmiregistry` application in any directory you want? Why?

3. Take a look in the **hello2** directory. It shows a different way of using the registry and a mechanism to create a registry object directly in the server code. What are the changes?

4. The **calculator** directory shows another example of RMI. In this case we are running a remote object that computes simple arithmetic results. There is one detail that students should notice in this example: the division operation can raise an exception (derived from the base `Exception` class), when the user requests a division by zero. We show this use in the client application. Students should also notice that the problem occurs only in the client. The server is unaffected and can respond to many different clients all causing the same problem. This means that RMI can and should use the standard exception mechanism.

5. Study the example **hello_remote**. You should implement an application that reads an `int`, a `double` and a `String` from the keyboard. Then you should create an object, put that data inside the object and send the object to the Server through a method called `publish(...)`. The server should print the object content when it receives it.

6. Study the example **hello_callback**. What happens when you launch a second client at the same time? What would be the necessary changes in the code to give support for multiple clients at the same time? Please implement those changes.